

Conversational AutoML - An Agentic Dataset Analyst

EN.705.625.8VL - Final Project Report

Ahnad Sayad
Johns Hopkins University
asayad1@jh.edu

Abstract — This work presents an agentic AutoML system for tabular data that combines large language models with graph-structured orchestration to deliver iterative, conversational data analysis. The objective is to enable non-expert users to pose high-level questions about a dataset (e.g., “What predicts GPA?”) and receive both trained model output and interpretable, natural-language insights without manually configuring pipelines or models.

At its core, the system uses a nested architecture built with LangGraph. An inner AutoML graph performs multi-iteration modeling: it profiles the dataset, lets an LLM select the target variable and task type, decides whether to apply PCA, proposes generic feature engineering steps, and trains a small model zoo of linear, tree-based, and neural models. A feature critic agent evaluates performance, decides whether further feature engineering is warranted, and stops when additional iterations are unlikely to help. When PCA is disabled, the system also computes feature-level importance metrics to support interpretability.

An outer conversation-level graph maintains a long-lived state across user turns. A meta-orchestrator LLM decides whether new questions can be answered from existing results or require a fresh AutoML run, enabling follow-up queries such as clarification, “what-if” questions, or changing the prediction target. Initial experiments on educational tabular data demonstrate that the system can automatically identify sensible targets, select appropriate tasks, achieve reasonable baseline performance, and provide coherent, human-readable explanations of which features matter and why.

I. INTRODUCTION

Machine learning on tabular data remains a cornerstone of real-world analytics across various domains. Yet the end-to-end workflow required to transform a raw dataset into actionable insight is still prohibitively complex for many users. Even experienced practitioners must navigate decisions about problem framing, data cleaning, feature engineering, model selection, hyperparameter tuning, and evaluation, each step requiring both technical expertise and iterative experimentation. Traditional AutoML systems attempt to streamline this process, but they often prioritize predictive performance at the expense of transparency and flexibility. Most operate as black-box optimizers: they generate pipelines, but provide little insight into why certain choices were made, how features contribute to predictions,

or how results relate to the user’s original analytical question. Moreover, they offer limited support for conversational refinement or follow-up questions that reinterpret or reframe the modeling problem.

Recent advances in agentic AI, provide an opportunity to rethink how AutoML is structured. Tabular modeling is intrinsically iterative, context-sensitive, and decision-rich, making it an ideal candidate for an agentic approach. Different stages of the process benefit from specialized reasoning: determining the appropriate target and task requires semantic understanding of the user’s question; proposing feature transformations requires pattern recognition across column types; deciding whether further iteration is worthwhile requires performance-aware evaluation; and delivering a final explanation requires accessible, natural language grounded in the results. A graph-structured, modular agent architecture allows these roles to be separated yet coordinated, enabling pipelines that adapt dynamically to both the dataset and the user’s intent.

This project explores whether hybrid agent architectures that blend LLM-based reasoning with structured retrieval, preprocessing, and model training can improve adaptability, interpretability, and user experience in tabular AutoML. There project aims to achieve three goals: (1) to determine whether agentic orchestration can automatically infer modeling intent, including target selection, task type, and preprocessing decisions, from natural language queries; (2) to evaluate whether iterative, LLM-guided feature engineering and critique can enhance model performance or interpretability relative to static pipelines; and (3) to demonstrate that maintaining conversational state across user turns enables more flexible and context-aware interactions, allowing follow-up questions to be answered without restarting the entire modeling process.

Together, these aims assess the potential for agentic AI to transform tabular data modeling from a rigid batch process into a collaborative, conversational analytic workflow.

II. BACKGROUND & RELATED WORK

A. PEAS Framework Analysis

We can characterize the system that Conversational AutoML uses by decomposing the problem its addressing into the PEAS framework:

Performance Measure:

Quantitative success measures within the system are evaluated through predictive accuracy, R^2 when it comes to regression tasks, cross-validation accuracy. These measures give the agentic system a measure of how good its derived/engineered features are at predicting a target from the dataset.

Qualitative success measures include interpretability of results, quality of natural language explanations, and alignment with the user’s queries. Since these are subjective to the user’s preference in response behavior, there are no metrics we can use to analyze them objectively. However, we can keep track of internal consistency by ensuring that the final user-facing output does not hallucinate any data that doesn’t exist in the environment it can observe.

Environment:

The environment includes tabular datasets, user-generated questions, and computational constraints. It evolves as new features are engineered and as conversation history accumulates.

Actuators:

The system acts on its environment by engineering features, updating dataset representations, training models, and producing natural-language textual analysis. These actions meaningfully alter both the internal state and are aligned with aiding the user’s understanding of the dataset provided.

Sensors:

Perception occurs through dataset profiling, schema inference, performance metrics, and natural-language input from the user. These signals guide the agent’s decisions at every step.

B. Agent Architectures

The agentic Conversational AutoML system presented here aligns closely with hybrid architectures. It contains reactive components such as schema profiling and deterministic preprocessing, as well as deliberative components such as LLM-driven target selection, PCA decisions, and feature engineering planning. By decomposing the modeling workflow into purpose-specific sub-agents connected through a structured graph, the system allows for controlled reasoning, tool use, iteration, and reflection, embodying core principles from the agent literature.

C. Environment Analysis

The Conversational AutoML system functions within a virtual analytic environment shaped by dataset characteristics, user queries, and LLM-driven tool interactions. This environment is neither static nor fully known; instead, it evolves as the agent performs feature engineering, trains models, updates internal state, and engages in multi-turn conversations.

The environment is fundamentally stochastic. Although certain elements such as dataset profiling, preprocessing pipelines, and cross-validation evaluation are deterministic given fixed inputs, many core transitions depend on the outputs of large language models. These include selecting the modeling target, determining whether the task is classification or regression, deciding whether PCA should be applied, proposing feature transformations, and judging whether additional iterations would improve performance. Since LLM outputs vary across identical prompts due to probabilistic sampling, repeated runs with the same dataset and question may follow slightly different orchestration trajectories.

The system operates in a partially observable environment. It can directly sense the dataset and inferred schema, missingness patterns and feature types, model performance metrics, and natural-language queries from the user. However, it can be prone to ambiguities in user intent beyond what is explicitly stated, and does not have access to the internal state of the LLMs guiding decisions. Because crucial aspects of the state space remain hidden, the system must rely on iterative refinement, heuristic reasoning, and adaptive feedback.

The system is fundamentally multiagent. The orchestrator, feature engineer, feature critic, analysis generator, and more all collaborate together to achieve the same goal. The environment only ever interacts with one of these agents, however. Therefore, the environment is best characterized as a collaborative multi-agent environment, where the system’s internal design exhibits multi-agent decomposition for functional clarity and control.

The environment is inherently sequential. Each decision such as adding engineered features, adjusting preprocessing settings, selecting models to train, or determining whether to initiate a new AutoML run has persistent consequences for all subsequent steps. There is no episodic reset between actions or iterations; instead, the system maintains continuity through a cumulative transformation history, evolving model performance records, iterative feedback from feature critics, and a persistent conversation state that influences how new queries are interpreted. This sequential property is especially prominent at the conversation level, where previous user questions and generated answers influence how future questions are addressed based on feedback mechanisms.

D. Feedback Mechanisms

The agentic AutoML system achieves its goals through a combination of supervised, unsupervised, and implicit evaluative feedback loops, each contributing to different phases of decision-making. At the modeling level, the system relies on supervised learning, using scikit-learn classifiers and regressors trained on labeled tabular data. Cross-validation accuracy provides concrete performance signals that the feature critic and orchestrator agents interpret as indicators of modeling success. This supervised feedback is central to determining whether newly

engineered features meaningfully improve predictive performance and influences decisions about whether to continue iterating.

In addition to supervised signals, the system incorporates elements of unsupervised feedback. Steps such as PCA rely on internal variance-capture criteria, and engineered features are created without labels, guided instead by structural patterns in the schema (e.g., column types, missingness, etc.). This unsupervised structure shapes the agent’s reasoning about which transformations may add value before any model is trained. The LLM-based feature engineer and critic also operate under a form of implicit evaluative learning, where they infer likely improvements from schema patterns and prior iteration outcomes rather than from explicit reward functions.

E. *Rationality & Decision Making*

As we learned in the class, rational agents select actions that maximize expected performance. Conversational AutoML does this through modular reasoning. The orchestrator rationally chooses targets and modeling tasks that best answer the user’s question. The feature engineer rationally proposes transformations that could increase predictive signal. The feature critic rationally evaluates marginal gains and determines whether further iterations are justified. The conversation-level meta-orchestrator rationally decides whether the agent should reuse knowledge or initiate a fresh AutoML run. These decisions demonstrate bounded rationality, where choices optimize performance within computational and informational constraints.

F. *Learning & Adaptation*

Across iterations and conversation turns, the system demonstrates adaptive behavior. We can see this in many places of the system. For example, feature choices evolve based on prior successes or failures. PCA usage adjusts according to interpretability demands inferred from the user. Conversation state enables incremental understanding of the analytical context in a way that mimics online learning. Throughout the entire system, modeling outcomes refine future decision making, creating a feedback loop between perception and action.

G. *Tooling*

This project used Langgraph as its agentic framework. Portkey and Ollama were used as the LLM backend for cloud inference, and local inference respectively. Sklearn and Pandas were used for training basic machine learning models on the provided dataset. The reasoning model used for the agents was GPT-OSS-120b.

III. SYSTEM DESIGN & METHODS

A. *Overall Architecture*

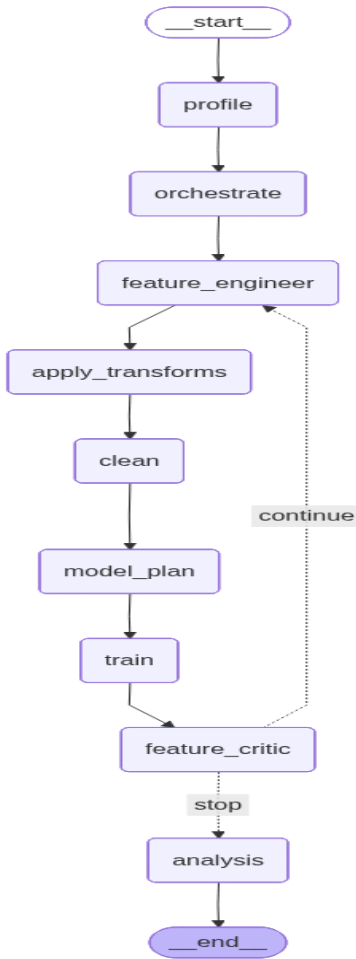
The system is built as a two-level agentic architecture using LangGraph. There exists an inner AutoML Graph

level, and an outer Conversation Graph level. The inner AutoML graph is a multi-agent workflow that profiles the dataset, chooses the modeling setup (determines the target, whether we are modeling classification or regression, and whether to use PCA), and produces a final natural-language analysis that is then shown to the user. The outer Conversation Graph is a conversation-level controller that decides whether to reuse previous AutoML results or run a new AutoML pipeline. It is also responsible for maintaining long-lived conversational and model state across user turns. The Conversation Graph layer is the only one exposed to the user. The result is a system that behaves like a conversational AutoML agent that can repeatedly answer questions about a dataset while remembering what it has already learned.

B. *Inner AutoML Graph: Agents, Tools, & Roles*

The AutoML graph is the heart of the agentic system. It consists of four different agents and five different tools. The AutoML graph starts with a *profiler* tool. This tool analyzes the dataset to identify column types, missingness patterns, and uniqueness counts. It provides structured metadata schema (a profile) that informs the downstream agents’ decisions. The profile generated from the profiler is then passed to the *orchestrator* agent. The orchestrator agent interprets the user’s question and the structure of the dataset. It determines which column should be predicted, whether the task is classification or regression, and whether dimensionality reduction should be used. Its decisions guide the direction of the entire AutoML process. Once the orchestrator determines the target, dataset structure, and intent behind the user’s query it passes the information along to the *feature engineer* agent. The feature agent proposes general, domain-agnostic feature transformations such as ratios, sums, missingness indicators, or text-derived features that could improve the model’s ability to detect signal, without relying on dataset-specific assumptions. This is important because it ensures that the AutoML process can adapt to different domains, and not dataset specific features. After the feature engineer agent determines what features to derive, it calls the *transformation* tool. This tool applies all derived transformations specified from the feature engineer agent and constructs a new dataset with said derived features. The augmented dataset is then passed to a *cleaner* tool. The cleaner attempts to minimize deficiencies within the existing dataset by cleaning and handling improper data for numeric and categorical variables. This includes imputing, scaling, and one-hot encoding the data. It also applies dimensionality reduction if the orchestrator specifies so. It then passes the cleaned, augmented dataset into the *model planner* tool. This tool selects a small but diverse set of candidate models appropriate for the determined task type, ensuring that the system explores different modeling philosophies. The chosen models were logistic regression, decision trees, and a small multi-layer perceptron for classification tasks, and linear regression, decision tree regressors, and a small multi-layer perceptron for regression tasks. Once the models and their hyperparameters were selected, the information would be passed to the *trainer* tool. The trainer tool fits the planned models using 3-fold

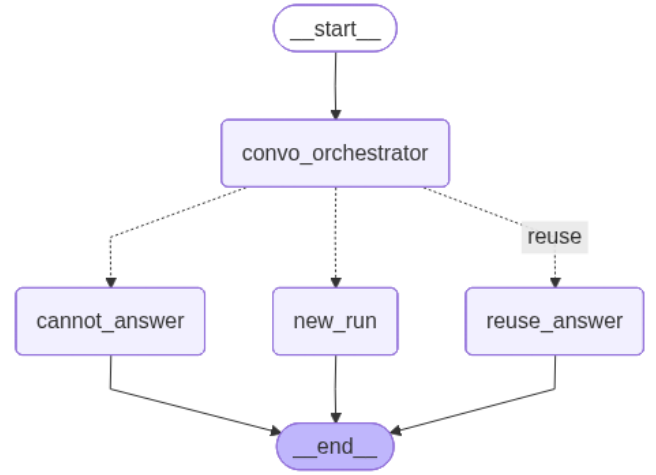
cross-validation, computes performance metrics, and derives feature importances when the model type permits. The performance measure outputs (as well as the features that the models trained on) are then passed to the *feature critic* agent. After each modeling iteration, this agent examines model performance, the existing features, and the transformations applied. It decides whether further feature engineering is likely to yield improvements and suggests additional transformations when it deems an improvement is possible. This process is then repeated three times, or until the feature critic deems any improvement from feature engineering is unlikely. Finally, all of the previous outputs are passed to the *analyst* agent. This agent synthesizes the results of all iterations using model scores, feature importances, and transformation history and produces a final explanation that directly answers the user’s question without revealing internal reasoning.



C. Conversation-Level Graph

The outer Conversation graph acts as a router; it determines whether or not the user’s query can be answered, and if it is answerable, can it use the previous results it has seen to provide a compelling argument to the user’s query. If it cannot, it decides to create a new AutoML run (which points to the internal AutoML graph that actually performs the AutoML process). The graph stores a state which consists of the entirety of the previous AutoML run, and all

question-answer pairings generated by the internal AutoML graph across all runs.



The mechanism of which it uses to decide whether it can use previous results or not is the *Conversation Orchestrator* agent. This high-level agent determines whether a new question can be answered using previous AutoML results or whether a fresh modeling run is required. It enables efficient multi-turn interaction by reusing results when possible.

If the conversation orchestrator agent determines that we can use a previous result to answer a user’s query, then it passes all relevant past result information to the *Previous Results Explainer Agent*. This agent analyzes the previously engineered datasets and trained model results with respect to a different question asked by the user, if the workflow decides that reusing a previous run is sufficient to answer a user’s question. If neither reuse or new runs are sufficient to answer the user’s question, the agent returns an explanation as to why.

Combined, the inner AutoML graph and the outer Conversation Graph orchestrators drive an end-to-end AutoML system that answers questions on natural language text from a user prompt.

D. Reasoning Process

The reasoning loop has three main stages: *Planning*, *Reflection*, and *Adaptation*. During the planning stage, the orchestrator agent plans the modeling configuration (target, task, PCA). The feature engineer agent plans candidate transformations. The model planner chooses small, interpretable models relevant to the task.

During the reflection stage, the feature critic evaluates the last iteration’s results and decides whether additional effort is justified. If the user asks another question, the conversation orchestrator agent reflects on whether the new question can be answered from existing results

During the adaptation phase, new features are added based on previous outcomes. PCA is turned on/off depending on interpretability needs for the training iteration. And if the results and derived features aren’t getting closer

to answering the user’s question, then the system changes its strategy across iterations.

E. Knowledge & Uncertainty Handling

The system’s knowledge is composed of several complementary sources: the structured information extracted from the dataset (such as schemas, feature types, and model evaluation metrics), the evolving conversational context maintained in memory, and the priors encoded within the large language model itself. Dataset profiling and schema inference provide concrete, deterministic knowledge about column characteristics, missingness patterns, and feature distributions. Meanwhile, the conversation state retains prior user questions, generated answers, and the results of previous AutoML runs, allowing the agent to ground new reasoning steps in historical context. These structured forms of knowledge are augmented by the LLM’s implicit domain expertise, which guides high-level decisions such as selecting modeling targets, proposing feature transformations, determining PCA usage, and synthesizing narrative explanations.

Uncertainty within the system arises from several sources and is handled through a blend of probabilistic and rule-based mechanisms. Model uncertainty is captured indirectly through cross-validation scores and their variability, enabling the agent to judge whether additional feature engineering is warranted or whether a model’s performance is stable. Uncertainty in LLM outputs is more subtle, as language-model reasoning is inherently stochastic and not accompanied by explicit confidence estimates. To mitigate this, the system constrains LLM outputs through structured prompts, JSON schemas, and limited action spaces, ensuring decisions remain grounded in the actual dataset. When LLM responses deviate from expected formats, fallback parsing strategies and guardrails prevent invalid states. Thus the agent effectively handles uncertainty through iterative refinement, constrained reasoning, and reliance on reproducible supervised-learning metrics. This combination allows the system to navigate imperfect information while maintaining robustness and interpretability.

F. Learning Components

Learning within the system occurs at two distinct but complementary levels: supervised model learning and behavioral adaptation through iterative agentic control. At the core of the AutoML loop, traditional supervised learning is used to train and evaluate predictive models using scikit-learn classifiers and regressors. Cross-validation provides performance estimates that guide decisions about whether engineered features are meaningful, whether PCA is beneficial, and which model families best capture structure in the data. These supervised signals serve as the primary quantitative feedback mechanism for assessing progress toward the modeling objective.

Above this, the system exhibits a form of structural and behavioral learning driven by repeated interaction between the LLM-based agents and the evolving modeling state. Although the language models themselves are not

fine-tuned or updated online, their behavior adapts across iterations as they receive different schemas, performance metrics, and transformation histories. This enables dynamic refinement of feature-engineering strategies, adjustments to preprocessing choices, and informed decisions about when to terminate the modeling loop. Similarly, at the conversation level, the meta-orchestrator agent learns how to respond to new user queries by referencing prior AutoML runs and previously generated explanations. The resulting adaptation is one where the agent refines its decision-making policy through state-conditioned reasoning. Together, these learning components allow the system to improve performance, enhance interpretability, and support multi-turn analytical workflows without requiring explicit reinforcement learning or continuous model retraining.

G. Experimental Setup and Environment

To evaluate the capabilities of the Conversational AutoML agentic system, we conducted experiments across a range of tabular datasets and interaction scenarios that mirror typical real-world analytical workflows. Although there is no single benchmark governing the system’s development, we tested its performance and usability on several standard datasets commonly used in machine learning education and Kaggle competitions, including the Titanic Survival dataset, the Adult Income dataset, the Housing Prices dataset, and the Student Grades dataset. For each of these tabular datasets, we would prompt the Conversational AutoML system with a natural language query that highlights the need for data augmentation and feature engineering of the dataset in question. The goal of such prompting was to examine the system’s iterative feature engineering, PCA decision making, and interpretability when PCA was disabled.

For example, for the Titanic dataset we would prompt it with queries such as “Did solo travelers have a higher survival rate than families?” to let the feature engineering agent derive features that quantify sibling count and fit the new features to survival rate. On the Student Grades dataset we would ask questions such as “Is parent support a strong predictor of a child doing well?” to let the feature engineer construct features that capture parental relationships with their children and then fit an ML model against the student’s GPA.

Beyond predictive accuracy, a key focus of the experimental setup was evaluating how the system behaved in interaction loops. These are the multi-turn dialogues in which users asked sequential questions about the same dataset. This tested the conversation orchestrator’s ability to decide when to reuse results from previous runs versus when to trigger a new AutoML pipeline. The system was assessed on its ability to maintain internal state, support clarifying or follow-up questions, and provide coherent, context-aware explanations grounded in prior modeling history. Collectively, these experiments illustrate the system’s effectiveness not only in achieving reasonable predictive performance but also in enabling flexible, conversational analytic workflows across diverse tabular environments.

IV. EXPERIMENTS & EVALUATIONS

A. Experimental Tasks and Benchmarks

To evaluate the performance and behavior of the agentic AutoML system, we conducted experiments across several representative tabular datasets and user-interaction scenarios. These included: Titanic Survival Rates, Housing Prices, Student Grades, & Adult Income. These four datasets represent classification and regression tasks, some of which where feature engineering is more important than others (feature engineering isn’t very important for the Housing Prices dataset). These datasets provide diverse structures and difficulty levels, making them suitable for validating both predictive performance and agentic reasoning capabilities.

B. Evaluation Metrics

The evaluation metrics chosen were listed previously in the PEAS framework analysis. As a reminder they are accuracy for classification tasks, and R^2 for regression tasks. The system uses 3-fold cross-validation to estimate predictive performance for both of these tasks. These scores provide stable performance estimates under limited computational budgets. We also include standard deviation across CV folds as a stability metric. This is because variation in model scores reflects robustness to dataset partitioning and informs the feature critic’s decisions about whether additional iterations are likely to help.

The qualitative evaluation metrics we use are feature interpretability quality and conversational effectiveness. When PCA is disabled, the system computes feature importances, which are later synthesized into natural-language explanations. We qualitatively assessed the coherence, grounding, and consistency of these explanations. Conversational effectiveness mainly measures whether or not the Conversation Orchestrator agent successfully reused previous results and avoided unnecessary recomputation.

C. Comparisons and Ablations

Conversational AutoML system’s performance were as follows on the datasets mentioned above:

TABLE I. CONVERSATIONAL AUTOML RESULTS

Dataset	Performance	
	<i>Highest Accuracy / R^2</i>	<i>std</i>
<i>Titanic Survival</i>	0.8159	0.0210
<i>Housing Prices</i>	0.9169	0.0038
<i>Student Grades</i>	0.8852	0.0029
<i>Adult Income</i>	0.7123	0.6131

As shown, the model performs exceptionally well on regression datasets (housing prices, student grades), and performs decently on classification datasets (titanic, adult income). In these tasks, the agentic system successfully inferred the correct ‘target’ column that the datasets were created to measure.

The model performed slightly worse on user queries where the target differed across runs. Full results of this behavior can be shown in the logs within the GitHub directory. This is as expected, as the datasets were created with the intention of modeling a specific behavior / feature (such as survival with the Titanic dataset, or house prices within the House Prices dataset). Thus creating and running AutoML systems that choose different targets than what the dataset intended slightly degrades behavior, however it often still yields valuable insights into the discriminative power of the features within the dataset.

D. Behavioral Observations and Emergent Dynamics

Across experiments, the system exhibited several emergent behaviors characteristic of coordinated multi-agent reasoning, despite being implemented as a structured collection of modular LLM-guided components.

First, the orchestrator consistently demonstrated effective planning behavior, correctly identifying the modeling target and task type from natural-language questions and adjusting its PCA decisions based on whether the user query implied a need for interpretability. The iterative interplay between the feature engineer and feature critic produced a meaningful refinement loop: the feature engineer proposed transformations informed by schema patterns and prior performance, while the critic evaluated marginal gains and curtailed further iterations when additional complexity was unlikely to improve results. This dynamic created a self-regulating modeling cycle that balanced exploration of new features with practical stopping criteria.

Another notable behavior was the system’s interpretability awareness. When questions referenced “why,” “importance,” or causal-like relationships, the agent reliably disabled PCA, thereby ensuring that downstream feature-importance computations and explanations remained transparent. At the conversation level, the conversation orchestrator displayed contextual continuity, correctly determining when new queries could be answered using previously trained models versus when a new AutoML run was required. This coordination across agents and conversation turns illustrates the system’s capacity for adaptive, goal-directed behavior, even in the absence of explicit reinforcement learning or centralized global planning.

E. Limitations

Despite its promising performance, the system exhibited several limitations and failure modes that highlight the challenges of integrating LLM-driven reasoning with automated machine learning pipelines. The most common issue involved minor hallucinations, where the LLM occasionally inferred nonexistent columns, fabricated small

numeric values, or overinterpreted weak statistical patterns when generating explanations. Although structured prompting and schema-constrained JSON outputs reduced these occurrences, they were not eliminated entirely.

The feature engineering agent also displayed limited expressive power: while it could apply generic transformations such as ratios, sums, missingness indicators, or regex-based text extraction, it lacked the ability to generate more sophisticated domain-driven features and occasionally proposed transformations that were syntactically valid but semantically irrelevant. Relatedly, the system sometimes produced incorrect or brittle regex patterns, leading to poorly populated or meaningless engineered categorical fields.

On the computational side, certain models, especially the multi-layer perceptron, could incur noticeably longer training times, especially when run repeatedly across iterations, which introduced latency in the user experience. These limitations illustrate the tension between flexibility and reliability in LLM-guided agentic systems and suggest opportunities for future improvements in guardrails, caching strategies, richer feature-generation primitives, and more efficient model selection mechanisms.

V. RESULTS AND ANALYSIS

Across the four evaluated datasets, the agentic AutoML system achieved strong baseline performance while exhibiting behaviors consistent with adaptive, goal-directed reasoning. On the Titanic Survival dataset, the best model reached an average accuracy of approximately 0.816 with a standard deviation of 0.021, closely matching common hand-tuned baselines. For Housing Prices, the system achieved an R^2 of about 0.917 with very low variance ($\text{std} \approx 0.0038$), indicating that the selected preprocessing and model choices captured a stable, high-fidelity mapping from features to target. On Student Grades, the system obtained an accuracy of roughly 0.885 with $\text{std} \approx 0.0029$, again highlighting both good performance and robustness across folds. The Adult Income results were more mixed: the system reached about 0.712 accuracy but with a very large standard deviation (≈ 0.613), suggesting unstable model behavior or inconsistent splits, and revealing a case where the agent’s decisions did not converge to a reliably strong configuration.

Manual inspection of the augmented datasets produced by Conversational AutoML yielded mixed results. The feature engineering agent tends to consistently add in “missingness” features that indicate the lack of a feature across a dataset when across most of the datasets it had a very low feature importance. Across other runs (such as in the housing dataset) when there aren’t many columns to work with the system would create nonsensical features to try such as “population per bedroom” on the housing dataset. In such instances it should have been clear the feature engineering agent that the proposed feature wouldn’t make much sense but it still attempted to do so just to see if it can extract marginal gains. Otherwise, the engineered features did their job and assisted in answering the user’s query.

These outcomes shed light on how the agentic design shapes system behavior. On datasets like Titanic, Housing, and Student Grades, the orchestrator’s decisions about target selection, task type, and PCA usage, coupled with the feature engineer and critic loop, appear to guide the system toward sensible, stable pipelines. The low variance on Housing and Student Grades suggests that the agent’s deliberative planning (for example, consistent preprocessing strategies and appropriate model family choices) pays off in terms of reliability. In contrast, the instability on Adult Income highlights the limits of bounded rationality: even with cross-validation, limited iterations and stochastic LLM decisions can lead to divergent pipelines, underscoring that agentic reasoning does not guarantee optimality in more complex or noisy environments.

The results also illustrate key trade-offs between reactivity and deliberation. The system’s deliberative components (orchestrator, feature engineer, critic) enable richer reasoning and adaptation, which improves interpretability and performance on several datasets, but at the cost of increased decision latency due to multiple LLM calls and iterative training. The more reactive components (e.g., fixed preprocessing and model evaluation) provide fast, deterministic feedback but cannot, on their own, adjust to user intent or dataset nuances. In practice, the hybrid architecture behaves as a boundedly rational agent: it makes reasonably good decisions under computational constraints, but occasionally “overthinks” (e.g., unnecessary iterations or complex transforms on Housing Prices) or “misfires” (e.g., unstable performance on Adult Income).

From an application perspective, these findings suggest that agentic AutoML is particularly well suited to domains where interpretability, flexibility, and conversational interaction are as important as raw accuracy, such as educational analytics, early-stage exploratory analysis in policy or health, and decision support for non-expert analysts. The system’s ability to respond to natural-language questions, adapt its strategy across iterations, and reuse prior results across user turns makes it a compelling analytic collaborator rather than a one-shot black-box optimizer. At the same time, the observed limitations like hallucinations, fragile feature engineering, and occasional instability underscore the need for stronger guardrails, better calibration of LLM decisions, and more principled uncertainty handling before such systems can be deployed in high-stakes environments. Overall, the experiments show that embedding classical ML pipelines within an agentic, multi-component architecture meaningfully changes the character of AutoML from a static optimization problem into a dynamic, interactive reasoning process.

VI. CONCLUSION AND FUTURE WORK

This work presented an agentic Conversational AutoML system that integrates classical machine learning pipelines with a multi-agent, LLM-driven reasoning architecture. By combining dataset profiling, iterative feature engineering, model training, and conversational synthesis within a structured LangGraph workflow, the system demonstrates how agentic design principles can enhance adaptability, interpretability, and user-centered interaction in tabular data analysis. Experiments across multiple datasets showed that

the system achieves competitive predictive performance, delivers coherent natural-language explanations, and maintains contextual awareness across conversation turns. More importantly, the results highlight how agentic features such as planning, reflection, and adaptation enable the system to behave not merely as a pipeline executor but as an analytical collaborator capable of responding intelligently to user intent.

Several lessons emerged from this exploration. First, LLM-guided orchestration can meaningfully improve usability and flexibility in AutoML but introduces stochasticity and occasional hallucinations that require careful constraint and validation. Second, iterative, LLM-driven feature engineering offers modest performance gains, suggesting potential for richer transformation libraries and more domain-aware reasoning. At the same time, challenges such as unstable performance on certain datasets, fragile regex transformations, and the computational overhead of repeated LLM queries reveal the need for more robust decision mechanisms and computational optimizations.

Looking forward, several extensions could deepen the system's capabilities. Future iterations might incorporate more sophisticated feature engineering tooling, expanded model selection, or meta-learning strategies that tailor orchestration decisions to dataset characteristics. Integrating explicit uncertainty handling rules could reduce hallucinations. On the conversational side, the system could benefit from richer memory structures or a retrieval-augmented knowledge base that allows it to accumulate insights across sessions.

Finally, ethical, safety, and interpretability considerations are pivotal as agentic systems become more autonomous. While the system aims to provide transparent, dataset-grounded analyses, LLM hallucinations and overconfident explanations pose risks in sensitive contexts. Ensuring that outputs clearly differentiate between empirical findings and model-derived inferences is essential. This consequences could be severe in cases where users take the data and act as if its "ground truth" instead of a hallucination. Overall, this work demonstrates the promise and responsibility of combining agentic architectures with machine learning automation, offering a foundation for more intelligent, adaptive, and trustworthy analytical tools.

REFERENCES

- Kaggle. (2012). *Titanic – Machine Learning From Disaster*.
<https://www.kaggle.com/competitions/titanic>
- Kaggle. (2020). *Housing Prices Dataset*
<https://www.kaggle.com/datasets/yasserh/housing-prices-dataset>
- Kaggle, Rabie El Kharoua. (2023). *Student Performance Dataset*
<https://www.kaggle.com/datasets/larsen0966/student-performance-data-set>
- Kaggle, UCI Machine Learning Repository. (2016). *Adult Census Income*
<https://www.kaggle.com/datasets/uciml/adult-census-income>